BACnet

BACnet communication protocol

Supported device types and versions

BACnet communication protocol (**B**uilding **A**utomation and **C**ontrol **Net**works) implements ANSI/ASHRAE 135-2001 standard.
This implementation was tested on the following devices:

- Siemens
  - Desigo PXM20 (Control unit, LON interface, BACnet over LON)
  - Desigo PXC22 (Control station, LON interface, BACnet over LON)
  - Desigo PXC22-E.D (Control station, Ethernet interface, BACnet/IP)
  - Desigo PXG80-N (BACnet router, Ethernet interface, LON interface, BACnet/IP, BACnet over LON)
- Delta Controls
  - DSC-1212E (System controller, Ethernet interface, BACnet/IP)
  - DAC-633 (Application controller, MS/TP interface connected to DSC-1212E, which works as BACnet router)
  - DAC-633 (Application controller, MS/TP interface connected to Moxa 5250 serial/ethernet converter and communicated directly as BACnet MS/TP device in UDP mode)
  - DAC-1146 - (Application controller, MS/TP interface connected to DSC-1212E which works as BACnet router)
- Sauter
  - EYK220F001 (Automation station, Ethernet interface, BACnet/IP)
  - EYR203F001 (Universal controller connected to EYK220F001)
  - EYR207F001 (Universal controller connected to EYK220F001)
- York
  - BACnet MS/TP MicroGateway (Communication card for York coolers, RS485 interface, BACnet MS/TP)
- SE-Elektronic GmbH:
  - E-DDC3.1 (DDC automation station, Ethernet interface, BACnet/IP)
- Klimasoft
  - BACnet/Mbus converter MBG-MSTP: Converter from BACnet to Mbus protocol (RS485 interface, BACnet MS/TP)

Characteristics of the current version:

- Communication in Ethernet (BACnet/IP) and LONTalk networks.
- Limited support of MS/TP network (master-slave token-passing on RS-485): without automatic searching of Master stations.
- Support of BACnet router (connection between BACnet/IP and LONTalk networks).
- Reading and writing of simple values (binary, integer, real, strings, date, time, etc..) and any ASN sequences.
- Support of polling method of data reading (messages ReadProperty-Request and ReadPropertyMultiple-Request).
- Support of change method of data reading (an optional registration by SubscribeCOV-Request or SubscribeCOVProperty-Request and next processing of ConfirmedCOVNotification-Request and UnconfirmedCOVNotification-Request).
- Writing the values by WriteProperty-Request.
- Dynamic change of I/O tag address by TELL command SETPTADDR (to read the values of Schedule objects).
- Work with objects of Schedule type (schedules).

BACnet protocol consider all the participants of the communication as the network devices. Each network device contains at least one (mostly just one) object *Device* (its Object Identifier must be unique in the whole network). This object contains other object of defined types (Analog Input, Analog Output, Analog Value, Binary Input, Binary Output, Binary Value, Calendar, Command, Event, Group, File etc.). The object detection - see a description of Request type Who-Is and Who-Has.
Each object has the properties, which can be required and optional. Moreover, each producer of BACnet devices can implement other properties when necessary.
The messages in BACnet protocol relates to the manipulation with objects and their properties. They are defined by the help of ASN.1 (Abstract Syntax Notation version 1) and encoded by simple version of BER (Basic Encoding Rules - encoding of ASN.1 messages).
The messages contain, besides fixed defined items, also the items 'Abstract Syntax & Notation'. It means that any sequence (or "tree"), which meaning is defined by an implementator, can be on the given place in the message. When using BER, it enables parsing of the message with the unknown items.
BER defines two basic item types (tags): application and context.
The application tags are predefined:

- **Null** - empty value
- **Boolean** - yes/no
- **Unsigned** - positive integer
- **Signed** - integer
- **Real** - 4-byte real number
- **Double**- 8-byte real number
- **Octet String** - sequence of character
- **Character String** - charset + text string
- **Bit String** - sequence of bits
- **Enumerated Value** - enumerated value
- **Date** - date
- **Time** - time
- **Object Identifier** - identifier of object (32-bit number, it consists of 10-bit number - Object Type and 22-bit number - Instance)

The context tags depend on the context (on the position in the message). Without knowing the context (a description of message that is being parsed), you can find out that on the particular position is the context tag No. 5 of the length 4 bytes, but you need an additional information whether the value is Unsigned, Signed, Real, Bitstring or other one.
Besides the simple application and context tags, the properties may be also complex:

- **Sequence** - the sequence that consists of other properties (both simple and complex one), they are either required or optional
- **Sequence of** - the sequence of N-tuple of properties
- **Choice** - one of N possibilities

Example - a dump from trace file of KOM process with the debug enabled:

```
=== ASN Body beg ===
objectIdentifier (tag 0) OBJID 0 analog-input,10
listOfResults (tag 1) SEQUENCE {
  propertyIdentifier (tag 2) ENUM 85 present-value
  propertyValue (tag 4) SEQUENCE {
   ENUM 1
  }
}
=== ASN Body end ===
```

Explanation:
It is the Sequence of two tags: *objectIdentifier* is contextual tag with the number 0, of Object Identifier type. Its value is Object type=0 (analog input), Instance=10.
The tag listOfResults contains the contextual tag 1 which is Sequence of two tags. First one is *propertyIdentifier*. It is the contextual tag No. 2, Enum type, value = 85 that correspond to "present-value". A second tag is contextual one, No.4. It the sequence that contains one Enumerated Value tag with the value=1 (application tag).
To parse this message, KOM process must know ASN.1 definition of message. Without it, the process can find out that the message contains the contextual tag 2 (value=1 byte) but cannot know that it is Enumerated Value. It is not able to interpret this byte (it could be Enumerated Value, Unsigned or Signed number) and do not know that the name of this contextual tag is *propertyIdentifier* and the value 85 correspond to "present-value".

The properties of objects are mapped on I/O tags in D2000 configuration. Due to the existence of contextual tags, you may specify an Application tag in the I/O tag. It determines the interpretation of the contextual tag. The parameter Complex address defines "a path" in the parse "tree" to get the values from the sequence that is defined by implementator.

Communication line configuration

- Communication line category: **TCP/IP-UDP**, LonWorks, Serial, SerialOverUDP Device Redundant.
- TCP/IP-UDP parameters:
  - Host: IP address or of network interface that is used for communication by KOM process. A symbolic name that can be translated to an IP address can be entered too.
    Note: a symbolic name **ALL** can be entered - in which case all available interfaces are used.
  - Port: UDP port number that is used for communication by KOM process (according to standard 0xBAC0, i.e. 47808).

Line protocol parameters

Keyword	Full name	Meaning	Unit	Default value
DBGI	Debug Input	Debug information about the input data. Meaning of the bits: • 1. bit - debugging of ASN message parsing • 2. bit - debugging of the I/O tag names that received a new value • next bits - not used	-	0

DTQ	Debug Timeout Queue	Debug information about messages in time queue.	-	False
DI	Device Instance	Non-zero value causes that KOM process answers to Who-Is request by I-Am message. It contains a defined Device Instance. Zero value causes that Who-Is request is ignored.	-	0
RB	Receive Buffer	(only for TCP/IP-UDP line) Size of received buffer which is set on UDP socket. Zero value means the buffer size remains unchanged. 8192 bytes is a normal size in Windows. If there are more stations or more intensive communication, the buffer should be enlarged.	bytes	0
RO	Receive Only	If value is True, any messages are not sent to any station on the line. This parameter may be used when listening the communication LonTalk: Configure the address, which is the same as the address of existing LonTalk device, on the line. Also configure the station with the device address which communication you need to listen to. The communication between devices is recorded to the log file of line. RO=True ensures that KOM process does not influence the communication by its commands and responds.	-	False
SC	Send Count	(only for LonWorks line) The retry count of one packet - default value is 1. However, in some situations when using *iLON (tm)10 Ethernet Adapter*, the first message did not pass and the communication started to work correctly when SC=2. **Note:** Later we found out that this was caused because Free topology bus had not been ended by a terminator. However, this parameter had been already implemented.	-	1
SD	Send Delay	(only for LonWorks line) A complement to *SC* parameter that defines a delay (v ms) after each sending of packet.	ms	0
VI	Vendor ID	Parameter *Vendor ID* of *I-Am* message (see the parameter *Device Instance*).	-	1

Line protocol parameters specific for BACnet MS/TP

Keyword	Full name	Meaning	Unit	Default value
BR	MS/TP baud rate	Baud rate of line. This parameter recalculates some timeouts that are defined in a **bit time** in the communication line protocol. Bit time is a multiple of period which is required when transferring 1 bit at the particular baud rate.	bits /sec	9600
MIF	MS/TP $N_{max\_info\_frames}$	Maximum of information frames that may be sent by KOM process before it must send a token. The standard does not specify the particular value. It recommends that the value must be 1 if this value is not configurable in a device. The higher value is set, the less time remains for other Masters. But on the contrary, it reduces the number of frames without information.	-	5
MO	MS/TP N_{min_octets}	Minimum number of data (bytes) received on the line to be received by KOM process before it indicates the line as "active".	-	4
MY	MS/TP my address	Address of KOM process on the line RS-485. The valid value is from the interval 0 - 127. It must be different from the addresses of other devices on the line (their addresses are defined in the station configuration).	-	1
TFA	T_{frame_abort}	Minimum time (the unit is length of bit transmission, i.e. it depends on MS/TP baud rate), after it expires, the whole frame is discarded if it does not receive other token. According to standard, the value may be higher but it cannot exceed 100 ms in the absolute time.	bits	60
TNT	T_{no_token}	Time (in milliseconds). After it expires, without receiving any data, the token disappears.	ms	500
TR	$T_{reply\_timeout}$	Minimum reply time of station.	ms	255
TS	T_{slot}	Time during which the station can generate a token.	ms	10
TU	$T_{usage\_timeout}$	Minimum time for which KOM process must wait while a partner start to use a token or respond on *Poll for master* frame. A standard value is 20 ms. According to a standard, the value may be higher - maximum 100 ms.	ms	20

Communication station configuration

The communication station corresponds to a device on BACnet network with which KOM process communicates.

- **Station type:** BACnet/IP station must be configured on TCP/IP-UDP line. LonWorks station must be configured on LonWorks line. MS/TP station must be configured on SerialOverUDP Device Redundant or Serial line.
- **Address:**
  - BACnet/IP station: IP address of station (in the form A.B.C.D, e.g. 172.16.0.99)
  - LonWorks station: address of LON subnet and LON node (in the form *subnet.node*, a subnet is 8-bit number and a node is 7-bit number)
  - MS/TP station: number of node on the line (0-254, address 255 is a broadcast)
- **Port:** (only for BACnet/IP): UDP port number on station (according to standard 0xBAC0, i.e. 47808)
- **Domain:** (only for LonWorks): 0 or 1, it relates with the line configuration. On LonWorks line you can configure a membership to one or two domains. On BACnet station, if you choose some domain, it means that the device belongs to this domain (it influences 'domain' bit in LON address).

- **Source network:** source network number (i.e. a network with KOM process). This parameter may not be set for LonWorks line. For TCP /IP-UDP line, it is 16-bit number (or it is not set, see Note 2).
- **Destination network:** 16-bit number of a destination network (i.e. a network including the device which communicates with KOM process).
  Set this for LonWorks line if KOM process communicates with the device that is placed after BACnet router. In that case, Address of station is the address of BACnet router and Destination address is the address of destination device.
  For TCP/IP-UDP line, use **Destination network** if there is a communication between different BACnet networks.

  Note 1: This configuration was tested as follows:
  - Line: *TCP/IP-UDP*
  - Station type: *BACnet/IP*
  - Address: *172.16.99.1* (address of BACnet router PXG80-N)
  - Destination network: *1*
  - Destination address: *1.1* (address PXC22 on LON network after BACnet router)

  KOM process communicated with the device PXC22 which was connected to LON network by BACnet router PXG80-N. KOM process communicates with BACnet router over Ethernet, so the line is TCP/IP-UDP. The communication between BACnet router and station PXC22 was done over LON network.

  Note 2: We tested the similar configuration. We used Delta Controls DSM-RTR (connected over Ethernet network) and Klimasoft MBG device (gateway on M-Bus) after it connected over MS/TP interface. The communication started if only *Destination network* (value 50020) and *Destination address* (value 96) were configured and not *Source network*. However, in other configuration, the communication proceeded also with the parameter *Source network*. We recommend you to try various settings of network parameters for the devices.

- **Destination address:** It is the address of destination device if KOM communicates with it over BACnet router. When setting this parameter, you can (but you may not, see note about E-DDC3.1) set also the parameter Destination network. Parameter **Destination address** should be in the form *subnet.node* (if destination device is in LON network) or in the form A.B.C.D (if destination device is in BACnet/IP network).

  Note 1: On BACnet/IP station you can configure **Destination address** in the form subnet.node (e.g. 1.31). This configuration corresponds to BACnet router, which communicates with KOM process over BACnet/IP and is connected to destination device over LONTalk network.

  Note 2: On BACnet/IP station you can configure **Destination address** as a number from the interval 1-255. This configuration corresponds to BACnet router, which communicates with KOM process over BACnet/IP and is connected to destination device by MS /TP bus (DAC-633).

  Note 3: On BACnet/IP station you can configure **Destination address** as bigger number (e.g. 2001), which works for E-DDC3.1.

- **Resubscribe interval:** Time in seconds. After it elapses, a station again gets a request to send changes of I/O tags. This parameter relates to the I/O tags with Request type that is equal to SubscribeCOV or SubscribeCOVProperty.
- **Max APDU:** Maximum size of message (APDU = Application Protocol Data Unit) that is sent by KOM process. A default value is:
  - 1467 octets for TCP/IP-UDP line
  - 487 octets for SerialOverUDP Device Redundant or Serial lines (BACnet MS/TP)
  - 55 octets for LonWorks line (the limitation depends on size of packets which may be transmitted over Ethernet and LonWorks. For LonWorks the maximum value is 206. The value 55 is due to the limitation of iLON 10 Ethernet adapter)

  The changing of default value is important for testing and adjusting to the stations which are able to process only smaller messages. For example, the reducing of *Max APDU* influences only the size and amount of messages ReadPropertyMultiple-Request. These messages are intended for a periodic reading of I/O tag value (see I/O tag configuration).

  Note: The setting of Max APDU does not affect the size of max-APDU-length-accepted in APDU BACnet-Confirmed-Request-PDU, with the help of which KOM process informs a partner what big message is capable to process. This parameter is configured by the station protocol parameter Segment-Response.

- **Priority:** A priority of message in BACnet protocol. There are 4 priorities: Normal (default), Urgent, CriticalEquipment and LifeSafety.
- **Rpt_timer & reply:** (only for LonWorks) The parameters Repeat timer (default value = 1) and Retry (default value =1) of LonTalk protocol.
- **Tx_timer:** (only for LonWorks) Parameter Tx_timer in LonTalk protocol. Default value = 3.
- **Timeout and retry:** A timeout in milliseconds to confirm the message. Default value according to BACnet protocol is 3000 ms. After the timeout elapses, the message is sent **retry**-times. If any confirmation is not received, an error count will increase on station.

  Note: When testing Siemens PXC64-U device (the communication over LonTalk), we had to set Retry=8, Timeout=300 (more retries with shorter timeout). Due to that, we had to increase the values COM_ERR=10, HARD_ERR=20 so that the station did not switch to error state at retrying to send message.
- **COM_ERR:** The value of error counter on station when the station switches to COM_ERR status. The situation when station does not reply on call for reading or writing of value could be consider as error. A negative confirmation of a command (refusal of recording) is not the error. Default value = 5. See the parameters *Timeout and retry.*
- **HARD_ERR:** The value of error counter on station when the station switches to HARD_ERR status. Default value = 10. See the parameters *Time out and retry.*
- **Register-Foreign-Device, R-F-D Time to live:** In this example the station is placed on LONTalk network after BACnet router which communicates with KOM process over Ethernet (e.g. Desigo PXG80-N). BACnet router sends the broadcasts from LONTalk to Ethernet as UDP broadcasts. If distributing of UDP broadcasts is disabled or KOM process is placed in other segment of network than BACnet router (it does not receive any UDP broadcasts), you should mark off the option *Register-Foreign-Device* on the station. This will cause, KOM process will send the message Register-Foreign-Device to BVLC router (BACnet Virtual Link Control) after start. The message calls for the registration to FDT table

(Foreign Device Table) in router. The router sends the broadcasts in the form of UDP unicast (whose distribution is not limited to one segment) to the devices that are registered in FDT table. TTL - time in seconds (1-65535) is the parameter of message Register-Foreign-Device. It defines the expiration of registration that stops sending of UDP unicasts. That is why KOM process must ask BACNet router to register it before TTL expires. If there are more stations after BACnet router, just mark off *Register-Foreign-Device* on one of them.

Note 1: If router does not support BBMD functionality (BACnet/IP Broadcast Management Device), it replies to the message Register-Foreign-Device with the error code and does not send LonTalk broadcasts to KOM process in the form of UDP unicasts. In that case you must use other solution (the communication over iLon Ethernet Adapter, the placing of KOM process on the same segment in network on which is BACnet router, etc.).
Note 2: Router Desigo PXG80-N supports this functionality (tested). The control station Desigo PXC22-E.D does not support it probably (not tested yet).
Note 3: In case of Desigo devices, if the D2000 KOM process is on a different network segment than the Desigo device, this parameter must be checked at the station. Otherwie Who-Is and Who-Has queries won't work (and thus addressing by object's name), as responses to these queries are sent as UDP broadcasts which will not go through a router.

- **Master:** (only for MS/TP): The station is Master type. KOM process transmits a token to Master station which has the next larger address than is the address of KOM process (the parameter of line MS/TP address). If the addresses of all Master station are lower than addresses of KOM process, token is given to Master station with the lowest address. If any Master station has not been configured, KOM process supposes to be the only master and does not give any token. You should get the information about the type of station from a producer or device documentation.

  Note: The current version of protocol does not contain the automatic search of Master station. You can find more information in the section Comment on BACnet MS/TP implementation.

Example of station configuration on TCP/IP-UDP line:

- **Station type:** BACnet/IP
- **Adresa:** 10.0.0.1
- **Port:** 47808
- **Source network:** 1

Example of station configuration on LonWorks line:

- **Station type:** LonWorks
- **Address:** 1.15
- **Domain:** 0

Station protocol parameters

Key word	Full name	Meaning	Unit	Default value
RSD	Receive-send Delay	Delay between the receiving of the reply from station and sending next packet.	ms	0
SR	Segment-Response	A byte that contains Max Segs and Max Resp parameter (see the specification of BACnet protocol). Only some values from the 0-127 are permitted, which are specified by BACnet standard. KOM process considers the value 128 as default: • LonWorks line: set the value to 0x70 (more than 64 segments are accepted, maximum length of message is 50 bytes) • TCP/IP-UDP line: set the value to 0x75 (more than 64 segments are accepted, maximum length of message is 1476 bytes) • Serial and SerialOverUDP Device Redundant line: set the value to 0x73 (more than 64 segments are accepted, maximum length of message is 480 bytes)	-	128
TSU	Time-Synchronization UTC	The parameter is important only if the synchronization is enabled on the tab "Time parameters" in the configuration of station. If the parameter is True (default), the time synchronization is executed by the message UTCTimeSynchronization-Request (the synchronization in UTC). If the parameter is False, the time synchronization is executed by the message TimeSynchronization-Request (the synchronization in the local time). **Notes**: • We recommend you to use the synchronization in UTC, if it is supported in the device - you can avoid the problems when advancing the time. • The requirements for the time synchronization are the unacknowledged messages, i.e. the device will not send the answer neither if it supports the time synchronization nor if it does not support. • The time synchronization has been tested on Siemens PXC36-E.D (HW=V3.02). This device supports the synchronization in both UTC and local time. You can find out the current time and date as "property local-date(56)" and "local-time(57)" of the object of Device(8) type. From this object, you can find out also "property utc-offset(119)" which defines the offset of local time from UTC (in minutes, i.e. -60 is Central European Time) as well as "property daylight-savings-status(24)", which defines whether the device works in the summer time (when testing in September 2012, the value on the device was True). After the time synchronization, the values "local-date(56)" and "local-time(57)" have been changed.	-	True

I/O tag configuration

Type of I/O tags: **Ai,Ci,Di,TiA,TiR,TxtI,Ao,Co,Dout,ToA,ToR,TxtO**.

I/O tag corresponds to object property.

- **Request type:** Reading and writing of the object properties may be done in several ways:
  - **ReadProperty -** a periodical reading of object property as request-response. A period of polling is set on the tab *Time parameters* of station. The message *ReadProperty-Request* represents the request and the message *ReadProperty-Ack*

represents the response from the device. The periodic reading loads the network and is ineffective. That is why, if the device supports the sending of change data, we recommend you to use SubscribeCOV or SubscribeCOVProperty.
The message *ReadProperty-Request* is sent if checkbox Subscribe/read is ticked off.

- **ReadPropertyMultiple** - the functionality is similar to the previous parameter. Unlike ReadProperty, more object properties are sent in one request-response, so the communication is much more effective. The message *ReadPropertyMultiple-Request* repre sents the request and the message *ReadPropertyMultiple-Ack* represents the response from the device.
The message *ReadPropertyMultiple-Request* is sent if checkbox Subscribe/read is ticked off.
- **WriteProperty -** the message *WriteProperty-Request* writes the values. The parameter Application tag must be specified as well. If Subscribe/read is marked off, the recorded value is reread by the message ReadProperty-Request after writing.
- **SubscribeCOV** - reading of object value by change method. If the checkbox Subscribe/read is ticked off, after starting, KOM process send the message SubscribeCOV-Request which asks the device to send information about the change of object value. You can specify whether the device will send the confirmed notifications (message ConfirmedCOVNotification-Request) or the unconfirmed ones (UnconfirmedCOVNotification-Request). The confirmed notification requires the explicit confirmation from KOM (the message BACnet-SimpleACK-PDU), so it loads the network. However, the probability that the notification will lost is lower than if you would specify the unconfirmed notification (if the device does not receive the confirmation, it repeats the message).

  Note 1: Besides the dynamic registration by the message SubscribeCOV-Request, some devices can support also static one (it is saved in the configuration). So the registration is not required and the checkbox Subscribe/read may not be ticked off.
  Note 2: The registration can be sent in regular intervals (because of the potential failure of device supply). You can set this interval on the station - the parameter Resubscribe interval.
- **SubscribeCOVProperty** - the functionality is similar to *SubscribeCOV*. Moreover, you can specify *Property identifier* (so you can monitor also the changes of other object properties as values) and *Increment* - a size of increment which causes the change will be sent (i.e. dead zone).
This message - *SubscribeCOVProperty-Request* is sent.

  Note: The device Siemens PXC64-U did not support the parameter *Increment*.
- **WhoIs** - the identification message *Who-Is-Request* to detect the type of Device Object in a device. The message *I-Am-Request* is the response (it contains the fields iAmDeviceIdentifier, maxAPDULengthAccepted, segmentationSupported, vendorID). If the I/O tag is TxtI, this information is extracted to the value of I/O tag in a text form. After you identify the Device Object, you can configure I/O tag for reading the property *object-list* of this Device Object. If the device implements this property, it returns the list of identifiers of all objects which it contains. Then you can detect the properties of these objects (object-name, location, description, present-value ..)

  Note: For Siemens PXC64-U you must set the Array index and then read the property *object-list*. Array index=0 defines number of array elements, Array index=1 up to N enables the access to the individual elements.
- **WhoHas** - the identification message *Who-Has-Request* to detect the object name from the object identifier or vice versa. The response is the message *I-Have-Request* (it contains the fields: deviceIdentifier, objectIdentifier and objectName). The message *Who-Has-Request* is sent only once when initialization of I/O tag (or after the TELL command SETPTADDR). It is intended for the transfer between names and identifiers of objects.
The message Who-Has-Request will contain either name or identifier of object depending on whether the Address type has been configured as *Name* or *Object type+Instance* in I/O tag.
If Subscribe/read is ticked off, you can use the information from cache, which is much more faster than detection from the communication.
- **ReadWriteScheduler** - the message *ReadProperty-Requestna* is used for the request, the message *WriteProperty-Request* is used for write (it writes N pairs *time-value*). Th configuration is used for the reading and writing of objects of *schedule* type, see the article Scheduler in Siemens Desigo devices.
- **GetEventInformation**: a detection of objects that are in alarm or error state or they need to be acknowledged, see the section I nformation about events.
- **AcknowledgeAlarm**: the acknowledgement of alarms that have been loaded by the request GetEventInformation. See the section Information about alarms. I/O tag must be the text output (TxtO).

- **Address type:** Each object in BACnet protocol is addressed by Object identifier. When designing the application in Desigo system, the objects are represented by name, but the object address is not accessible and can vary following the changes of application. As regards Delta Controls devices, they contain the objects whose addresses are defined by the author of the application. For this reason, there are two ways how to define the address of I/O tag which corresponds to two *Address type*:
  - **Name:** enter the object name. A type of object and number of instance are found out dynamically from the communication. To avoid the overloading of communication lines when starting the KOM process, data are stored in a cache.
  - **Object type + Instance:** enter the type of object and number of instance. This is recommended for BACnet objects with the constant addresses.
- **Object type:** type of objects, whose properties will be read/written. You can use the predefined types or write the number of new type of object which has been defined by a producer. The type of object is 10-bit number.
- **Instance:** a sequence number of object within the object type. Each object has a unique Object Identifier in the device, which is a pair [Object type, Instance].
- **Object Name:** name of object, Address type= Name, i.e. it means the I/O tag address is detected dynamically from the communication. Object Name must be set without spaces in the beginning and end, and the uppercase and lower case letters must correspond to object name that is stored in the device with which it communicates.
- **Property type:** type of property - set only *PropertyIdentifier* for *Simple*, and both *PropertyIdentifier* and *Complex address* for *Complex*. The complex type of property is necessary when parsing by implementer of extension of the messages (Abstract Syntax & Notation). When sending the messages ReadProperty-Request, ReadPropertyMultiple-Request, SubscribeCOV-Request, SubscribeCOVProperty-Request, the *Complex address* is ignored.
- **Property identifier:** identifier of object property. You can use the predefined properties or number of new property which has defined the producer. The type of property identifier is Enumerated Value, the properties 0-511 are reserved for BACnet, the numbers from 512 to 4194303 can be used for the device producers.
- **Array index:** some properties may be defined as value array. The particular item of array can be read or written.
- **Application tag:** must be specified when writing the value (Request type=WriteProperty) and possibly for other types of requests, if parsed response contains the context tags which application type is unknown because it is the extension of messages defined by implementer. The exception is an output text tag that is considered to be 'AnyTree', as regards the unspecified application tag, and is

used to write any user-specified ASN sequence.

Note: If the the value is *Invalid*, it is not written as the defined *Application tag*, but as *Application tag "Null"*.

- **Complex address:** address of tag in a 'tree' in connection with the extension of messages that have been defined by implementer.
  Example of address: *[1].[3].2.1*
  Description:
  [1] - context tag No.1 (it is assumed that it is the sequence),
  [3] - it is the context tag No. 2 in this sequence (it must be the sequence),
  2 - it is the second tag in order in this sequence (it must be the sequence),
  1 - it is the first tag in order in this sequence.
  The address in 'tree' starts from the level propertyValue (see the examples below). The easiest way to view the parsed message is to turn on a debug on the line and watch the debug info on the console or in the log of line.

Example 1: There is a device that contains the object of type 2 (Analog Value) with the instance number 1. It is assumed that the device sends the object value as a triplet of numbers. The first number is the current value, the second one is one-minute average and the third one is ten-minute average. The log of parsed message could be the following:

```
=== ASN Body beg ===
objectIdentifier (tag 0) OBJID 2 analog-value,1
listOfResults (tag 1) SEQUENCE {
 propertyIdentifier (tag 2) ENUM 85 present-value
 propertyValue (tag 4) SEQUENCE {
  REAL 1.40000E+00
  REAL 1.10000E+00
  REAL 1.30000E+00
 }
}
=== ASN Body end ===
```

If you wants all three values, you must configure three I/O tags (Object type=analog_value, Instance=1, Property-identifier=present-value, Property-type=complex), which differ in the complex address (for the first I/O tag = 1, for the second one = 2, for the third one = 3). Tick off the checkbox Subscribe/read only in one configuration of these I/O tags, because the response to one request is the message with three values. When sending the messages ReadProperty-Request, ReadPropertyMultiple-Request, SubscribeCOV-Request, SubscribeCOVProperty-Request and WriteProperty-Request, the complex address is not used.

Note: If you configure the I/O tag with Property-type=simple, its value would be set on the first found value after parsing of the message (see the example, it is 1.40000E+00).

Example 2: Siemens Desigo PXC64-U contains I/O tag (Object type=schedule, Instance=6, Subscribe-read is ticked off, Property-identifier=weekly-schedule, Property-type=complex, Array index=1, Complex address=1). A debugging has been started on the line. After the I/O tag is saved, KOM process sends the request and writes the response:

```
=== ASN Body beg ===
objectIdentifier (tag 0) OBJID 17 schedule,6
propertyIdentifier (tag 1) ENUM 123 weekly-schedule
propertyArrayIndex (tag 2) UNSIGNED 1
propertyValue (tag 3) SEQUENCE {
 SEQUENCE {
  TIME 0:0:0.0
  UNSIGNED 2
  TIME 4:0:0.0
  UNSIGNED 3
  TIME 22:0:0.0
  UNSIGNED 1
 }
}
=== ASN Body end ===
```

In propertyValue there is the sequence of 6 values (time and positive number alternately). If you want to access to the first time, you have to set Complex address=1.1, if to the first positive number, set Complex address=1.2. I.e. the first element - sequence - the second element in the order within it (UNSIGNED 2). If you need to access to more times and/or values at the same time, you must configure several I/O tags and tick off the checkbox Subscribe/read only in one of them.

Note 1: The value of I/O tag remains Unknown, after creating and saving it with the complex address 1, because this address matches with 1. element in propertyValue, which is the sequence but not a simple typ.

Note 2: The I/O tag of *Text* type are able to cover not only simple value but also any ASN sequence. The values are written according to rules for record of ASN sequence. If you set Complex address=1 and change the I/O tag to text input or text output in the previous example, its value will be a string " T0:0:0.0; u2; T4:0:0.0; u3; T22:0:0.0; u1; ". If Property-type=complex, but Complex address is not defined, a result is "0{ T0:0:0.0; u2; T4:0:0.0; u3; T22:0:0.0; u1; }".

- **Increment:** increment of value change in the object property which causes the reporting of change (see SubscribeCOVProperty).
- **Confirmed:** if the checkbox is ticked, it specifies whether the device will send the confirmed notifications (ConfirmedCOVNotification-Request) or unconfirmed one (UnconfirmedCOVNotification-Request) for the configured Request types SubscribeCOV and SubscribeCOVProperty.
- **Subscribe/read:** if the checkbox is ticked, the proper messages for reading/registration of value changes are sent for the configured Request types:
  ReadProperty: the message ReadProperty-Request
  ReadPropertyMultiple: the message ReadPropertyMultiple-Request
  SubscribeCOV: the message SubscribeCOV-Request

SubscribeCOVProperty: the message SubscribeCOVProperty-Request
ReadWriteScheduler: the message ReadProperty-Request

- **Period from:** if the nonzero value is set and the checkbox Subscribe/read is ticked, the messages Subscribe/read will not be sent in the interval *P olling period*, which has been configured on the station, but in the defined period (in seconds). In this way you can configure various Polling periods for different objects on one station. Moreover, you can detect with the help of one I/O tag with Request type ReadProperty and small *Perioc* whether the station communicates.
- **Local time:** if the checkbox is ticked, the recorded respectively read times and dates are in a local time. If not, they are in monotonous UTC time.
- **Flags-to-flags:** if the checkbox is ticked, it causes that besides of I/O tag value, its user flags FA,FB,FC,FD are set for the Request types SubscribeCOV and SubscribeCOVProperty. The value *status-flags* (BACnetStatusFlags type) is mapped to these flags, if they are sent. BACnetStatusFlags is a quaternion of bits (in-alarm, fault, overridden, out-of-service) that support extra information about the object value.
- **Write priority:** for the record of 'commandable' properties, you can specify the priority 1-16. 1 = top priority, 16 = the lowest priority, 0 = none priority.

Browsing of address space

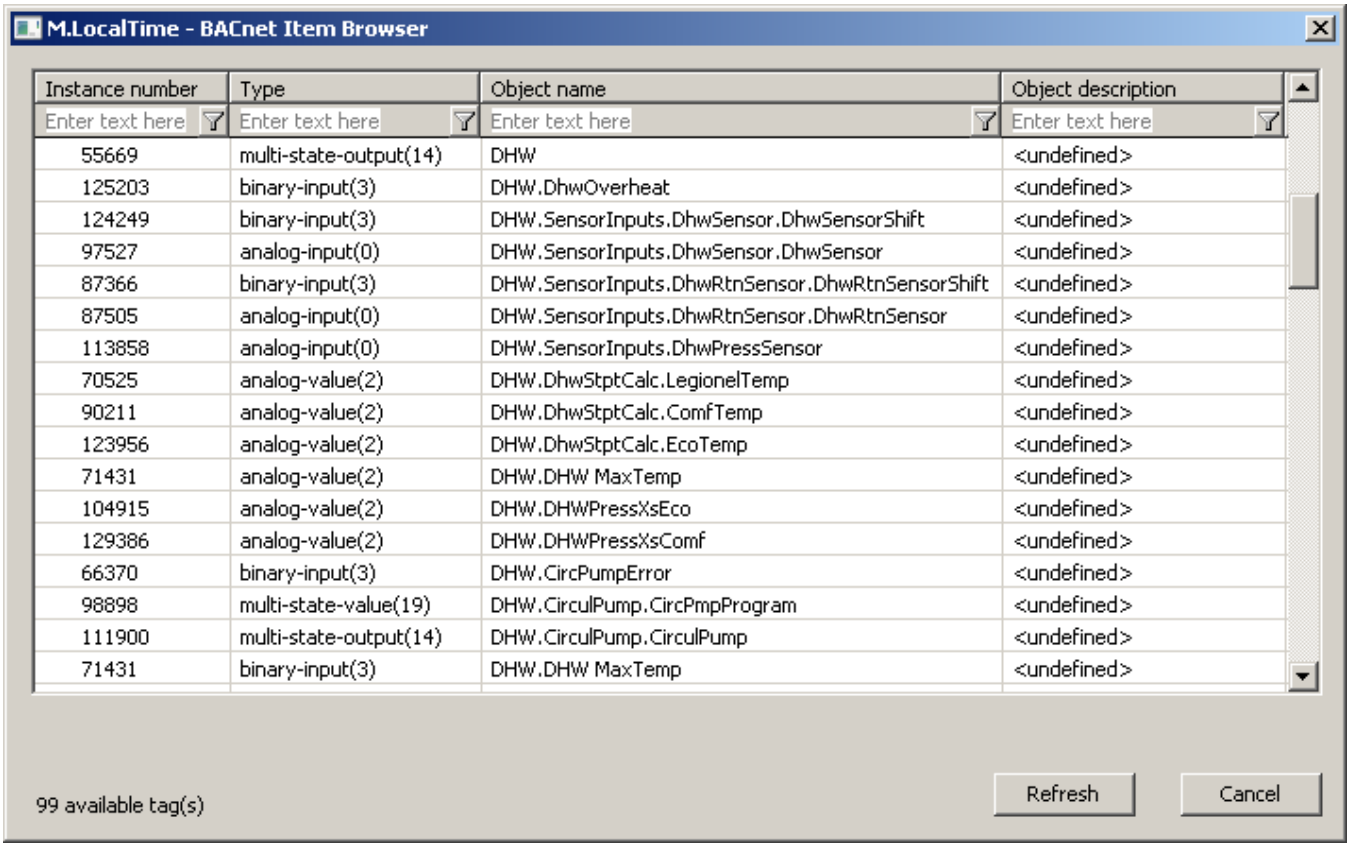
When clicking on "**Browse**" button in **Address** tab of I/O tag, **Bacnet Item Browser** dialog box opens. In this dialog box user may browse the address space of a device and insert its items to the address dialog of I/O tag.

The items can be filtered through the four basic criteria:

- Instance number
- Type
- Object name
- Object description

Note: In case of *Type*, *Object name* and *Object description* it is not necessary to enter the full text in filter field. Notation "*FILTERED EXPRESSION*" is supported. The symbol * represents any text before and after the expression.

Note: Using Ctrl+C it is possible to copy content of **Bacnet Item Browser** into Windows clipboard. All rows will be copied unless a specific row is selected.



Record of any ASN sequence
Any ASN sequence can be written with the help of I/O tag - text output (TxtO) without setting of application tag. The rules are the following:

- the element consists of the optional number of context tag, the letter defining the application tag and the value
- the application tags are written as follows (usage with and without context tag):
  - Null: *[tag] n*, example: " n", " 3n",
  - Boolean: *[tag] b [0|1|n|y|N|Y]*, example: "b0", " 3b1"

    - Unsigned: *[tag] u value*, example: "u 123", " 10 u123"
    - Signed: *[tag] s value*, example: "s-123", " 10s 5"
    - Real: *[tag] r value*, example: "r 1.23", " 10r-3.14"
    - Double: *[tag] d value,* example: "d 1.23", " 10 d -3.14"
    - Octet string: *[tag] O string*, every byte of string is written as hex number (byte 1 is 01, byte 26 is 1A), example: "O 1A33f0", " 10 Obb004E"
    - Character string *[tag] C 'string'*, example: "C 'hello world' ", " 10C 'apostrophe '' in the string' "
      The string must enclosed in the apostrophes. If the apostrophe occurs inside the string, it must be double (see the second example). Empty string can be set as follows: " C; "
    - Bit string: *[tag] B bits*, example: "B 100101", " 23B00101"
    - Enumerated value: *[tag] E value*, example: "E 123", " 10 E123"
    - Date: *[tag] D day.month.year[.day_of_week]*, example: "D 1.10.2005", " D3.4.2004.5" (Monday=1 .. Sunday=7)
    - Time: *[tag] T hour:minute:sec[.ms]*, example: "T 5:12:33.133", " T10:00:00"
    - Object identifier: *[tag] o type:instance* or *[tag] o objid*, (type is 10-bit number, instance is 22-bit number, objid is 32-bit number. The examples show a binary and single record of this application tag with type=3 (binary-input) and instance=2
      " o 3:2", " 3o3:2"," 3o 12582914"
- the sequence consists of the single elements separated by the blank spaces and/or semicolons, e.g. " 1b0 2u13 ; 3 B 1001;4E14"
- the sequence can contain the nested sequences
- the nested sequence begins with the optional number of context tag and character '{'. If any context tag is not entered, 0 is used.
- at the end of nested sequence is '}'
- the example of nested sequence: "1u2 2{ 1s-1; 2E0 }", two levels of nesting: " { 1{ u23 s34 } 2E56 3r7.89 }"

Note 1: If ASN sequence is a result of the reading of I/O tag from communication and I/O tag is Txt type, this ASN sequence is written into it (according to above mentioned rules) on conditions that:

- before each element is blank space and the element is followed by semicolon (e.g. " 1E4; 2B111; 3u1;"),
- between context tag and the letter is not blank space,
- between the letter and value is not the blank space,
- the context tag is before nested sequence (e.g. " 0{ 0o1:2; 1E4; }",
- format of time is hh:mi:ss.mss, e.g. " T11:01:02.000; 1T12:00:00.000; ",
- format of date is dd.mm.yyyy, e.g. " D25.01.2005; 3D01.01.2005;".

Note 2: The empty sequence may be recorded by the string " ", the string of length 0 (i.e. "") is ignored.

Examples of I/O tag configuration (Siemens Desigo PXC64-U device):
Analog input:

- Request type: SubscribeCOV
- Object type: analog-input(0)
- Instance: 1
- Confirmed: Y
- Subscribe/read: Y
- Property type: Simple
- Property identifier: present-value(85)

Binary input:

- Request type: SubscribeCOV
- Object type: binary-input(3)
- Instance: 1
- Confirmed: Y
- Subscribe/read: Y
- Property type: Simple
- Property identifier: present-value(85)

Binary value:

- Request type: ReadProperty
- Object type: binary-value(5)
- Instance: 8
- Subscribe/read: Y
- Property type: Simple
- Property identifier: present-value(85)
- Application tag: Enum

Note: If the *Application tag* is not set properly (for binary value it is Enum), Desigo station returns an error 'invalid-data-type' when trying to record it:

```
error-class ENUM 2 property
error-code ENUM 9 invalid-data-type
```

Siemens Desigo PXC64-U device requires to set the following parameters for *Application tag*:

- binary-value, binary-output: Enum
- multi-state-value, multi-state-output: Unsigned
- analog-value, analog-output: Real

Comment on input-output I/O tag: you can configure the I/O tag which is both input and output:
Ao - Analog output:

- Request type: ReadProperty
- Object type: analog-value(2)
- Instance: 45
- Subscribe/read: Y
- Property type: Simple
- Property identifier: present-value(85)
- Application tag: Real (this is necessary for the record of value)

When writing the value, the message WriteProperty-Request is used. As Subscribe/read is ticked, the written value is reread. If I/O tag would be configured with Request type WriteProperty, its behavior would differ only in an absence of periodic value reading (when starting KOM process and during its running, the period is set on the station, tab *Time parameters)*.

Comment on Siemens Desigo devices: I/O tags has a text name in the Desigo control system. The instance of I/O tag may be detected from the file DOTS00.DAT in the application configuration - it is placed 24 bytes before beginning of the name.

Scheduler in Siemens Desigo devices

D2000 supports the reading and writing of schedulers. A scheduler contains BACnet attributes *weekly-schedule(123)* and *exception-schedule(38)*. Weekly-schedule is the field of 7 items (one item for each day in week). Each item represents a sequence of time-value pairs that defines the value changes of scheduler in given time. When reading and writing you can configure also *Array index* which enables to access to the items in an array *weekly-schedule(123).* The array index is 1-7 for single days (1=Monday), the index 0 contains a size of array (value 7 for the attribute *weekly-schedule(123))*, value 0 up to N for *exception-schedule(38)*.
Exception-schedule is intended for the holidays that require different regime as is configured for common days. Exception-schedule is the sequence of 0 up to N items. One item always contains date (or range of dates), several time-value pairs (as regards weekly-schedule) and priority (1= top, 16= the lowest). The priority defines which of the items will be used if they overlap.

Reading of scheduler (the attribute weekly-schedule)

- Value after value: a dynamic change of complex address (1.1, 1.2, 1.3 etc.) in a script enables to read all values and times similarly as for other properties.
- All times and values for one day at the same time:
  - Value type: a text input (reading of scheduler) or text output (read/write)
  - Request type: ReadProperty (reading of scheduler), ReadWriteScheduler (read/write)
  - Subscribe/read: Y
  - Object type: schedule(17)
  - Instance: number of instance (e.g. 6) which is found from Desigo configuration or with the help of WhoIs request.
  - Property type: Complex
  - Property identifier: weekly-schedule(123)
  - Array index: 1 up to 7, it depends on the loaded day
  - Application tag: if it is not defined, Unsigned is used (it is necessary only when writing of value)
  - Complex address: 1 (address of sequence)

  The times and values separated by semicolon are loaded into the text value (see

Note 2

).

When recording the value of scheduler you should realize that the value can be sent with various application tags (Unsigned, Signed) but the device waits the particular tag (the easiest way to find it, is the loading the value when debugging on the line). The application tag of value is defined by the item Application tag in configuration. The valid value of Application tag can be Boolean, Unsigned, Signed, Real and Double. If other type is set, Unsigned value is automatically sent. A value type can be changed dynamically - if the first character of text value is B,U,S,R or D, it stands for (B)oolean, (U)nsigned, (S)igned, (R)eal or (D)ouble.

Recording of scheduler (the attribute weekly-schedule)

- You must configure Request type=ReadWriteScheduler and assign the sequence of time-value pairs separated by semicolon into I/O tag of text output (e.g. "0:0:0; 1; 2:30:40.5; 2; 5:00:00;1".
- The text string with the length=0 is ignored so that the "empty sequence" will not be recorded to the scheduler after saving the D2000 configuration or starting KOM process. For this reason, if the time plan of scheduler must be deleted for the particular day, the string of nonzero length (it contains neither time nor value: " ") must be assign to I/O tag.

Note: Another possibility, besides the special request ReadWriteScheduler, to write weekly-schedule is recording of ASN sequence, e.g. the sequence "{ T0:0:0 u1; T2:30:40.5 u2; T5:0:0 u1 }" corresponds to the value "0:0:0; 1; 2:30:40.5; 2; 5:00:00;1". The I/O tag configuration differs only in Request type=ReadProperty. You can also write the time frame for whole week, if Array index is not set and the value contains 7 sequences for the individual days, e.g. "{ T0:10:0 u3 T1:3:0 u1; } {T2:0:0.0 u2 T5:30:10.0; u3; } { T6:0:0.0 u2 T7:0:0.0 u3} { T20:0:0.33; u1} { T21:0:0.0; u1} { T22:0:0.0; u2} 0{ T0:0:0.0; u3; T1:2:0.0; u1; T2:0:0.0; u2; T5:30:10.0; u3}".

Recording of scheduler (the attribute exception-schedule)

The recording of the exception-schedule is supported over the recording of ASN sequence.
Example: I/O tag configuration:

- I/O tag type: text output (TxtO)
- Request type: WriteProperty
- Subscribe/read: Y
- Object type: schedule(17)
- Instance: 6
- Property type: exception-schedule
- Application tag: not defined (write the whole ASN sequence)

With the help of this string "0{ 0D2.10.2005 } 2{ T1:0:0; u1; T12:0:0; u3 } 3u10" you can write the time frame for the day October 2, 2005. The scheduler has the value 1 from 1:0:0, the value 3 from 12:00:00. The priority exception-schedule is 10. When rereading of value (Subscribe/Read was configured) with the activated debug on the line, you can see the following log:

```
=== ASN Body beg ===
objectIdentifier (tag 0) OBJID 17 schedule,6
propertyIdentifier (tag 1) ENUM 38 exception-schedule
propertyValue (tag 3) SEQUENCE 3 {
 readResult (tag 0) SEQUENCE 0 {
  date (tag 0) DATE 2.10.2005 NoDay
 }
 listOfTimeValues (tag 2) SEQUENCE 2 {
  time TIME 1:0:0.0
  value UNSIGNED 1
  time TIME 12:0:0.0
  value UNSIGNED 3
 }
 eventPriority (tag 3) UNSIGNED 10
}
=== ASN Body end ===
```

To set the exception-schedule for the range of dates, write the value "0{ 1{ D5.10.2005; D8.10.2005}} 2{ T1:0:0; u1; T7:0:0; u3; } 3u15". This value for the range of dates 5.10.2005-8.10.2005 sets the scheduler from 1:00:00 to the value 1 and from 7:00:00 to the value 3. The priority exception-schedule is 15. When rereading of value (Subscribe/Read was configured) with the activated debug on the line, you can see the following log:

```
=== ASN Body beg ===
objectIdentifier (tag 0) OBJID 17 schedule,6
propertyIdentifier (tag 1) ENUM 38 exception-schedule
propertyValue (tag 3) SEQUENCE 3 {
 readResult (tag 0) SEQUENCE 0 {
  dateRange (tag 1) SEQUENCE 1 {
   startDate DATE 5.10.2005 NoDay
   endDate DATE 8.10.2005 NoDay
  }
 }
 listOfTimeValues (tag 2) SEQUENCE 2 {
  time TIME 1:0:0.0
  value UNSIGNED 1
  time TIME 7:0:0.0
  value UNSIGNED 3
 }
 eventPriority (tag 3) UNSIGNED 15
}
=== ASN Body end ===
```

Several items of above mentioned types can be written to the exception-schedule by the string which contains jointed sequences, e.g. "0{ 0D2.10.2005 } 2{ T1:0:0; u1; T12:0:0; u3 } 3u10 0{ 0D3.10.2005 } 2{ T0:0:0; u2; T5:0:0; u3; T14:0:0; u1 } 3u11 0{ 1{ D5.10.2005; D8.10.2005}} 2{ T1:0:0; u1; T7:0:0; u3; } 3u15 "

Scheduler in Delta Controls devices

We also tested the reading and writing to the scheduler in Delta Controls DAC-1146 device. Only BACnet attribute *weekly-schedule(123*) was tested. The weekly-schedule is an array with 7 items (one item for each day in week). Each item is the sequence of time-value pairs that define the value changes of scheduler in given time. In contrast of Siemens Desigo devices, the schedulers are implemented with Boolean values True/False that are externally presented as Enum values 0/1. This is the example of scheduler Delta Controls:
"{ T0:10:0 E0 T1:3:0 E1; } {T2:0:0.0 E1 T5:30:10.0; E0; } { T6:0:0.0 E0 T7:0:0.0 E1} { T20:0:0.33; E0} { T21:0:0.0; E1} { T22:0:0.0; E0} 0{ T0:0:0.0; E0; T1:2:0.0; E1; }"
The writing has done and then reading of value E2, however the interpretation by DAC-1146 is unclear.

Information about events

The request GetEventInformation-Request is intended to call for the list of objects that are in the state *Offnormal* or *Fault* or whose change to *Offnormal*, *Fault* or *Normal* has not been acknowledged from the device.
Example of I/O tag configuration:

- I/O tag type: text input (TxtI)
- Request type: GetEventInformation
- Subscribe/read: Y
- Object type: undefined
- Instance: undefined
- Property type: complex
- Application tag: undefined

The message GetEventInformation-Ack is a reply to GetEventInformation-Request, which contains the list of objects and the list of properties for each object:

- objectIdentifier: identifier of object
- event-state: object status (0=normal, 1=fault, 2=offnormal, 3=high-limit, 4=low-limit, 5=life-safety-alarm)
- acknowledgedTransitions: 3 bits that define whether the last transition to the offnormal, fault or normal status was acknowledged
- eventTimeStamps: the time stamps of last transition to the offnormal, fault and normal status
- notifyType: defines whether it is a notification of alarm (0) or event (1)
- eventEnable: 3 bits that defines whether the events report to to-offnormal, to-fault, to-normal
- eventPriorities: 3 unsigned values defining the event priority

At the end of list, there is the attribute moreEvents that is non zero if the list of events is incomplete (e.g. the exceeding of length of maximum message, etc.). For this reason, you must reconfigure I/O tag and set Object type and Instance on the last object in the list. It causes the generation of new message GetEventInformation-Ack.

Example of response GetEventInformation-Ack:

```
=== ASN Body beg ===
listOfEventSummaries (tag 0) SEQUENCE 0 {
 objectIdentifier (tag 0) OBJID 0 analog-input,1
 event-state (tag 1) ENUM 3 high-limit
 acknowledgedTransitions (tag 2) BITSTRING 1,1,1 to-offnormal(0),to-fault(1),to-normal(2)
 eventTimeStamps (tag 3) SEQUENCE 3 {
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 9.11.2005 Wed
   time TIME 13:28:49.0
  }
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 9.11.2005 Wed
   time TIME 13:25:59.0
  }
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 9.11.2005 Wed
   time TIME 13:25:56.0
  }
 }
 notifyType (tag 4) ENUM 0 alarm
 eventEnable (tag 5) BITSTRING 1,1,1 to-offnormal(0),to-fault(1),to-normal(2)
 eventPriorities (tag 6) SEQUENCE 6 {
  eventPriority UNSIGNED 3
  eventPriority UNSIGNED 3
  eventPriority UNSIGNED 7
 }
 objectIdentifier (tag 0) OBJID 214,1
 event-state (tag 1) ENUM 2 offnormal
 acknowledgedTransitions (tag 2) BITSTRING 0,1,0 to-offnormal(0),to-fault(1),to-normal(2)
 eventTimeStamps (tag 3) SEQUENCE 3 {
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 18.11.2005 Fri
   time TIME 12:52:11.0
  }
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 18.11.2005 Fri
   time TIME 11:16:23.0
  }
  dateTime (tag 2) SEQUENCE 2 {
   date DATE 9.11.2005 Wed
   time TIME 12:23:58.0
  }
 }
 notifyType (tag 4) ENUM 0 alarm
 eventEnable (tag 5) BITSTRING 0,0,0 to-offnormal(0),to-fault(1),to-normal(2)
 eventPriorities (tag 6) SEQUENCE 6 {
  eventPriority UNSIGNED 3
  eventPriority UNSIGNED 3
  eventPriority UNSIGNED 7
 }
}
moreEvents (tag 1) BOOLEAN FALSE
=== ASN Body end ===
```

Information about alarms

You can acknowledge the events and alarms, whose list has been loaded by the request GetEventInformation. According to BACnet protocol, the format of this request is (the record in ASN.1 syntax):

```
********************** Confirmed Alarm and Event Services ******************
AcknowledgeAlarm-Request ::= SEQUENCE {
  acknowledgingProcessIdentifier [0] Unsigned32,
  eventObjectIdentifier [1] BACnetObjectIdentifier,
  eventStateAcknowledged [2] BACnetEventState,
  timeStamp [3] BACnetTimeStamp,
  acknowledgmentSource [4] CharacterString,
  timeOfAcknowledgment [5] BACnetTimeStamp
}
```

The more detailed description of the parameters is stated in a literature.
The acknowledgement is executed by the writing of above mentioned sequence to the text of output I/O tag according to the record of any ASN sequence.

Example: The loading of list of alarms and events with the help of GetEventInformation:

```
 ===  ASN Body beg ===
 listOfEventSummaries (tag 0) SEQUENCE 0 {
  objectIdentifier (tag 0) OBJID 0 analog-input,1
  event-state (tag 1) ENUM 4 low-limit
  acknowledgedTransitions (tag 2) BITSTRING 0,1,1 to-offnormal(0),to-fault(1),to-normal(2)
  eventTimeStamps (tag 3) SEQUENCE 3 {
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 11:54:23.0
   }
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 10:4:37.0
   }
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 11:54:23.0
   }
  }
  notifyType (tag 4) ENUM 0 alarm
  eventEnable (tag 5) BITSTRING 1,1,1 to-offnormal(0),to-fault(1),to-normal(2)
  eventPriorities (tag 6) SEQUENCE 6 {
   eventPriority UNSIGNED 3
   eventPriority UNSIGNED 3
   eventPriority UNSIGNED 7
  }
  objectIdentifier (tag 0) OBJID 0 analog-input,2
  event-state (tag 1) ENUM 3 high-limit
  acknowledgedTransitions (tag 2) BITSTRING 1,1,1 to-offnormal(0),to-fault(1),to-normal(2)
  eventTimeStamps (tag 3) SEQUENCE 3 {
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 10:41:59.0
   }
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 10:12:20.0
   }
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 10:12:21.0
   }
  }
  notifyType (tag 4) ENUM 0 alarm
  eventEnable (tag 5) BITSTRING 1,1,1 to-offnormal(0),to-fault(1),to-normal(2)
  eventPriorities (tag 6) SEQUENCE 6 {
   eventPriority UNSIGNED 3
   eventPriority UNSIGNED 3
   eventPriority UNSIGNED 7
  }
  objectIdentifier (tag 0) OBJID 214,1
  event-state (tag 1) ENUM 2 offnormal
  acknowledgedTransitions (tag 2) BITSTRING 0,1,1 to-offnormal(0),to-fault(1),to-normal(2)
  eventTimeStamps (tag 3) SEQUENCE 3 {
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 11:54:23.0
   }
   dateTime (tag 2) SEQUENCE 2 {
    date DATE 25.11.2005 Fri
    time TIME 10:12:20.0
   }
   dateTime (tag 2) SEQUENCE 2 {
```

```
      date DATE 25.11.2005 Fri
      time TIME 10:12:21.0
     }
    }
    notifyType (tag 4) ENUM 0 alarm
    eventEnable (tag 5) BITSTRING 0,0,0 to-offnormal(0),to-fault(1),to-normal(2)
    eventPriorities (tag 6) SEQUENCE 6 {
     eventPriority UNSIGNED 3
     eventPriority UNSIGNED 3
     eventPriority UNSIGNED 7
    }
   }
   moreEvents (tag 1) BOOLEAN FALSE   === ASN Body end ===
```

The object is the first in list
objectIdentifier (tag 0) OBJID 0 analog-input,1
which is in the status low-limit
event-state (tag 1) ENUM 4 low-limit
and contains unacknowledged transition to the offnormal status (i.e. according to D2000 terminology it is an active alarm *low-limit* which requires the acknowledgement):
acknowledgedTransitions (tag 2) BITSTRING 0,1,1 to-offnormal(0),to-fault(1),to-normal(2)
This transition occurred in time
dateTime (tag 2) SEQUENCE 2 {
date DATE 25.11.2005 Friri
time TIME 11:54:23.0
}
The alarm is acknowledged with the record of text value
0u1; 1o0:1 ; 2E2; 3{ 2{ D25.11.2005 T11:54:23} } 4C'D2000' 5{ 2{ D25.11.2005 T11:55:23 }}
and the individual items are as follows (see the definition of AcknowledgeAlarm-Request):

- *0u1u1* - tag 0, unsigned value 1 - acknowledgingProcessIdentifier = identifier of process which acknowledges the alarm (it could be optional)
- *1o0:1:1* - tag 1, the identifier of object of type 0, instance 1 - eventObjectIdentifier = the identifier of object which contains the alarm
- *2E2* - tag 2, enum value 2 - eventStateAcknowledged = the acknowledged value (BACnet standard defines the following events that can be acknowledged):
  normal (0),
  fault (1),
  offnormal (2),
  high-limit (3),
  low-limit (4),
  life-safety-alarm (5)
  acknowledge the transition to the offnormal status in this case
- *3{ 2{ D25.11.2005 T11:54:23} }* - timeStamp = the sequence with date and time which must be acknowledged (it must match to date and time which has been loaded)
- *4C'D2000'* - acknowledgmentSource = a source of alarm acknowledgement (obviously it is any string)
- *5{ 2{ D25.11.2005 T11:55:23 }}* - timeOfAcknowledgment = date and time when the alarm was acknowledged

After acknowledging of alarm and loading the list of alarms and events by the request GetEventInformation, you will see the alarm has been acknowledged. This record:
acknowledgedTransitions (tag 2) BITSTRING **0**,1,1 to-offnormal(0),to-fault(1),to-normal(2)

will change to:

acknowledgedTransitions (tag 2) BITSTRING **1**,1,1 to-offnormal(0),to-fault(1),to-normal(2)

If the object had been in *normal* state, it would not have been listed in the list of alarms. In this example it is in *low-limit* state, i.e. it is an active acknowledged alarm.

To load the list of alarms, their record to a browser and acknowledgment of alarm, you must write the service scripts which parse the text value of the request GetEventInformation and compound it for AcknowledgeAlarm.

Comment on address cache

If the station has at least one I/O tag with Address Type = Name, the numerical address is detected by *Who-Has-Request* from ObjectName when initializing these I/O tags. After getting the address, the data (ObjectName; Object Type; Instance; DeviceInstance) are saved to cache file and are available for next starting of KOM process.
For each station there exists one cache file. It is placed in an application directory, subdirectory Cache. Its name is Cache_*station_name*.txt, e.g. Cache_B. StationX1.txt.
When saving BACnet station, the data are reloaded from file. If the file was not found, the *Who-Has* messages are generated to all I/O tags that contain Address Type = Name in this station.
This behavior may be used after changing the configuration of the device which communicates with KOM process (if the addresses of objects was changed when changing the configuration). Just delete the cache file in this station and save it - the cache file occurs again and is filled with the acquired data.

Note: the interaction of cache and I/O tags with Request type = WhoHas.

- If the parameter Subscribe/read is marked off in the I/O tag, the cache is not searched and the *Who-Has-Request* message is sent to the communication. The obtained information is not written to the cache. It is important only for the objects that generate and vanish dynamically, which would overflow the cache.

- If Subscribe/read is not marked off, the cache is searched. If *ObjectName* or *ObjectIdentifier* was not found, the *Who-Has-Request* message is sent to the communication. If the respond comes, the information is written into the cache.

Comment on Delta Controls devices

The test configuration included DSC-1212E device connected to the local network Ethernet and DAC-633 device connected to DSC-1212E over MS/TP interface (RS-485). D2000 KOM communicated directly with DSC-1212E and DAC-633 over DSC-1212E (it performed the function of BACnet router). We used the software ORCAview 3.30 Build 1481 from which we detected the following configuration information:

- **DSC-1212E**
  I logged on to ORCAview and found *DSC-1212E* device. Then the object *BACnet Settings 3200* (3200 is a software address of the particular device) occurred in the right panel in **Navigator** dialog window. Clicking on *BACnet Settings 3200* opened the dialog window which contained several tabs. The *Setup* tab included also the port UDP/IP. After I had clicked on it, its parameters displayed in the bottom of the window, e.g. *Network*, *IP Address* and *UDP Port*.
  When communicating with DSC-1212E, in D2000 configuration of station I could set *Source network* to the value of *Network* or not. The parameters *IP Address* and *UDP Port* must be used to set the parameters *Address* and *Port* in the station configuration in D2000. A station type was *BACnet/IP.*

  Note: On the tab *Advanced* - BACnet Properties, the parameter *Local Netwok Number* was set on 10032 (it was the same as the parameter *Network* of the port *Ethernet* on the tab *Setup*). This port is used for communication BACnet over Ethernet (without use of IP protocol), which is not supported in D2000 yet.

- **DAC-633**
  I logged on to ORCAview and found *DAC-633* device. Then the object *BACnet Settings 3202* (3202 is a software address of the particular device) occurred in the right panel in **Navigator** dialog window. Clicking on *BACnet Settings 3202* opened the dialog window which contained several tabs. The *Setup* tab included also the port MS/TP with the attribute *Status*, the value *Active,* and the attribute *Status Reference* of value *BACnet Settings 3202 (NET1)*. After I had clicked on it, its parameters displayed in the bottom of the window, e.g. *Network* and *MAC Address*.
  When communicating with DAC-633 I had to set the following station parameters in D2000:
  - Station type: BACnet/IP
  - Address: the parameter *IP Address* in configuration DSC-1212E
  - Port: the parameter *UDP Port* in configuration DSC-1212E
  - Source network: the parameter *Network* in configuration DSC-1212E
  - Destination network: the parameter *Network* in MS/TP port in configuration DAC-633
  - Destination address: the parameter *MAC Address* in MS/TP port in configuration DAC-633

  As D2000 communicates with DAC-633 over DSC-1212E, the parameters *Address*, *Port* and *Source network,* which correspond to DSC-1212E, and *Destination network* and *Destination address*, which correspond to MS/TP network between DSC-1212E and DAC-633, must be set in the station configuration.

Both devices support the polling and change method of reading of data (both ReadProperty and SubscribeCOV).
The parameters *Object type* and *Instance* in the configuration of I/O tag was detected in ORCAview from the attributes *Object* and *Description* (e.g. object 3200.AI12 is Analog Input, Instance 12; the object 3202.PG3 is Program, Instance 3). The parameter *Address type* is *Object type+Instance* and *Request type* is *SubscribeCOV*, if it is not defined other parameter.

I/O tags:

- Analog-input
- Analog-output (Application tag: Real - possibility to write)
- Analog-value (Application tag: Real - possibility to write)
- Binary-input
- Binary-output (Application tag: Enum - possibility to write)
- Binary-value (Application tag: Enum - possibility to write)
- Calendar (Request type: ReadProperty, Property type: Complex, Property identifier: datelist(123), Application tag: Date, Complex address: 1,2,.. etc. - gradual reading from the array; possibility to write the whole calendar with the help of ASN sequence but Complex address is not entered, e. g. "0D1.9.2006; 0D3.10.2006; 0D8.9.2006")
- Event-enrollment
- Schedule (Request type: ReadProperty or WriteProperty - possibility to write; Property identifier: weekly-schedule(123))
- Program (Request type: ReadProperty, Application tag: Boolean - possibility to write - stop and start the program)
- Multi-state-value (Application tag: Unsigned - possibility to write)
- Trend-log (the cyclic buffer of values that are saved with the configured interval)
  - Property identifier: 1074 - array of trend time stamp in tens of milliseconds since a time of data reset
  - Property identifier: 1076 - bit string array (attributes of values?)
  - Property identifier: 1077 - array of trend values
  - Property identifier: 1105 - time of resetting

Comment on E-DDC3.1 devices

The communication was revived according to the following configuration:

- Line: *TCP/IP-UDP*
- Station type: *BACnet/IP*
- Address: *10.0.6.23* (IP address E-DDC3.1)
- Port: *47808*

- Source network: *not defined*
- Destination network: *not defined*
- Destination address: *2001* (this was acquired from BACnet OPC server - the parameter "Device ID")

We tested ReadProperty, SubscribeCOV, WriteProperty (in the objects of Binary Output, Binary Value, Analog Value type).
The defined I/O tags (the test configuration contained only these types):

- Analog-input
- Analog-value (Application tag: Real - possibility to write)
- Binary-input
- Binary-output (Application tag: Enum - possibility to write)
- Binary-value (Application tag: Enum - possibility to write)

The device (2.01.05) with the original firmware did not work with Who-Is. We had to upgrade this firmware to the version 2.01.16. Then everything, including WriteProperty, worked properly.

Comment on Siemens Desigo devices

Based on document "DESIGO INSIGHT: Installation, setup and communication - Engineering guide", the recommended address settings for LonWorks communication are:

- DomainID: 0x49
- SubnetID: 1
- NodeID:
  - 1..100 - range reserved for automation stations (PX) and system devices (BACnet routers)
  - 101..120 - operating devices and DESIGO INSIGHT management stations
  - 121..127 - temporary operating devices (e.g. the PXM20 operator unit) and tools (DTS)

Comment on Klimasoft MBG-MSTP devices

The test configuration contained Moxa NPort 5150 (a converter from UDP to RS485) connected to the converter MBG-MSTP, which communicated with a heat meter Siemens UH50-A21R-SK06-G. Only analogue inputs was detected. The converter MBG-MSTP supported only ReadProperty.
The communication was revived according to the following configuration:

- Line: SerialOverUDP Device Redundant with IP address and Moxa NPort 5150
- The parameter of BACnet protocol that are configured on the line:
  MS/TP address: 6 (any address 1-254 that does not clash with other addresses on RS485 bus)
  MS/TP N max_info_frames: 20. Default value is 5. If this parameter exceeds the number of I/O tags, it causes that all I/O tags are loaded in one cycle (as regards the periodical loading), which is not interrupted because of sending a token. We recommend you not to increase the value of parameter if other devices are connected with RS485 and they could have problems if they do not receive any token for a longer time.
  MS/TP usage_timeout: 99. According to a standard, the value must be under 100 ms. Default value 20 ms caused the problems with communication (MBG-MSTP did not manage to react till timeout).
- Type station: *MS-TP*
- Address: *1* (according to configuration MBG-MSTP)
- I/O tag configuration:
  Request type: ReadProperty
  Object type: analog-input(0)
  Instance: according to configuration MBG-MSTP or device which is behind it

Comment on iLON 10 Ethernet adapter

When using iLON 10 Ethernet adapter for communication you should realize that the communication processor of this device (Neuron 3120) has relatively short default buffers (network buffer is 66 bytes. Therefore it does not receive the longer packets. It can be seen in the web interface iLON 10 - tab *Status* and the parameter Missed Messages increases when trying to read the value (e.g. always after saving the I/O tag if its parameter Subscribe/read is marked). This problem occurred when schedule of scheduler contained more than 4 pairs *time-value*. When there were 4 pairs, the length of LON message was 63 bytes. After adding next pair via PXM20, the loading of schedule failed.

Warning! The utility Nodutil enables to increase the size of received packet by increasing of both the network and application buffer. However, if you set **im proper values, you can damage iLON 10 device** (Neuron 3120 stopped communication with the user processor).
If you decide to preset the size of buffers, just change it from 66 bytes to 88 (and reduce the number of buffers) because KOM process informs, in its packets, that it is able to receive 50-byte ASDU (+ BACnet header 3 bytes and LON header 16 bytes).
The buffers of PCLTA-10 ISA adapter (built on the communication chip Neuron 3150) had the long enough buffers (255 bytes).
After we bought the new iLON 10, the configuration was successful with these parameters:

- configuration software: *Echelon Node Utility Release 1.82*
- size and quantity of buffers:

```
DEVICE:0> (B)uffer configuration
Node buffer configuration
Type                      Count        Size        Bytes
Receive transaction       11        13        143
```

```
Transmit transaction        2        28        56
App buffer in                2        82        164
App buffer out                2        82        164
Net buffer in                5        82        410
Net buffer out                2        82        164
App buff out priority        1        82        82
Net buff out priority        1        82        82
==> Total bytes = 1265
```

Comment on BACnet MS/TP implementation

The implementation of BACnet MS/TP is not complete yet. It was tested only on basic configuration (KOM against BACnet MS/TP MicroGateway produced by York company). The communication works with both Serial line (RS485/RS232 converter was used) and SerialOverUDP Device Redundant line (Moxa NPort series 5xxx was used).
MicroGateway (York company) implements the communication of MS/TP Master type (i.e. it sends the frame *Poll for master* to detect the existence of other Master stations). KOM relies on a partner station and does not contain the method of sending of this frame. Also it does not solve a failure of the station to which it sends a token.
The actual implementation is applicable only for communication with one Master station. It can be applicable for communication with one or more Slave stations (not tested). Also the time ratio could cause some problems in loaded systems, i.e. KOM could answer to the frames *Poll for master* or *Token* later than in required time, which may cause the collisions on the line. The time ratios may become worse when you activate the logs on the line.

Literature

ANSI/ASHRAE Standard 135-2001: BACnet - A Data Communication Protocol for Building Automation and Control Networks

ASN.1 Complete by Prof. John Larmouth

Changes and modifications

-

Document revisions

- Ver. 1.0 - August 30, 2005 - first version
- Ver. 1.1 - October 20, 2005 - support of schedulers, reading and writing of ASN sequences
- Ver. 1.2 - November 22, 2005 - support of a dynamic detection of I/O tag address from object name, address cache in a file
- Ver. 1.3 - June 14, 2006 - support of BACnet router (tested with PXG80-N)
- Ver. 1.4 - April, 02, 2008 - a partial support of BACnet MS/TP protocol (tested on BACnet MS/TP MICROGATEWAY on the cooler produced by York company)
- Ver. 1.5 - June 25, 2010 - testing the cooperation with E-DDC3.1 from SE-Elektronic GmbH

Related pages:

Communication protocols